***Лабораторна робота №12***

***«Використання регулярних виразів у Windows PowerShell»***

***Регулярні вирази - призначення та використання***

Регулярні вирази (або скорочено "регекспи" (regexp, regular expressions)) володіють більшою силою, і здатні сильно спростити життя системного адміністратора або програміста. У **PowerShell** регулярні вирази легко доступні, зручні у використанні і максимально функціональні. **PowerShell** використовує реалізацію регулярних виразів .NET.

Регулярні вирази - це спеціальна міні-мова, службовець для розбору (parsing) текстових даних. З її допомогою можна розділяти рядки на компоненти, вибирати потрібні частини рядків для подальшої обробки, робити заміни і т. д.

Знайомство з регулярними виразами почнемо з більш простої технології, що служить подібним цілям - з групових символів. Напевно ви не раз виконували команду ***dir***, вказуючи їй як аргумент маску файлу, наприклад ***\*.exe***. В даному випадку зірочка означає "будь-яку кількість будь-яких символів". Аналогічно можна використовувати і знак питання, він буде означати "один будь-який символ", тобто ***dir ??.exe*** виведе всі файли з розширенням ***.exe*** і ім'ям з двох символів. У **PowerShell** можна застосовувати і ще одну конструкцію - групи символів. Так наприклад ***[a-f]*** означатиме "один будь-який символ від ***a*** до ***f***, тобто ***(a, b, c, d, e, f)***", а ***[smw]*** будь-яку з трьох букв (***s***, ***m*** або ***w***). Таким чином команда ***get-childitem [smw]??.exe*** виведе файли з розширенням ***.exe***, у яких ім'я складається з трьох букв, і перша буква або ***s***, або ***m***, або ***w***.

***Оператор PowerShell -match***

Для початку вивчення ми будемо використовувати оператор **PowerShell** ***-match***, який дозволяє порівнювати текст зліва від нього, з регулярним виразом справа. У разі якщо текст підпадає під регулярний вираз, оператор видає ***True***, інакше - ***False***.

***PS C:\> "PowerShell" -match "Power"***

***True***

При порівнянні з регулярним виразом шукається лише входження рядка, повний збіг тексту необов'язково (зрозуміло, це можна змінити). Тобто достатньо, щоб регулярний вираз зустрічався в тексті.

***PS C:\> "Shell" -match "Power"***

***False***

***PS C:\> "PowerShell" -match "rsh"***

***True***

Ще одна тонкість: оператор ***-match*** за замовчуванням не чутливий до регістру символів (як і інші текстові оператори в **PowerShell**), якщо ж потрібна чутливість до регістру, використовується ***-cmatch***:

***PS C:\> "PowerShell" -cmatch "rsh"***

***False***

***Використання груп символів***

У регулярних виразах можна використовувати і групи символів:

***PS C:\> Get-Process | Where-Object {$\_.Name -match "sy[ns]"}***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------ ----- ----- ----- ------ ----- -- ----------***

***165 11 2524 8140 79 0,30 5228 mobsync***

***114 10 3436 3028 83 50,14 3404 SynTPEnh***

***149 11 2356 492 93 0,06 1592 SynTPStart***

І діапазони в цих групах:

***PS C:\> "яблуко", "апельсин", "груша", "абрикос" -match "а[а-п]"***

***апельсин***

***абрикос***

У лівій частині оператора ***-match*** знаходиться масив рядків, і оператор відповідно вивів лише ті рядки, які підійшли під регулярний вираз.

Набагато цікавіше використовувати діапазони для визначення цілих класів символів. Наприклад ***[а-я]*** означатиме будь-яку букву російського алфавіту, а ***[a-z]*** англійського. Аналогічно можна чинити з цифрами - наступна команда виведе всі процеси, в іменах яких зустрічаються цифри:

***PS C:\> Get-Process | Where-Object {$\_.Name -match "[0-9]"}***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------- ----- ---- ----- ---- ------ -- -----------***

***57 2 404 1620 16 0,05 984 ati2evxx***

***110 4 2540 4868 36 0,20 852 hpgs2wnd***

***105 3 940 3292 36 0,19 2424 hpgs2wnf***

***91 3 2116 3252 34 0,06 236 rundll32***

Так як ця група використовується достатньо часто, для неї була виділена спеціальна послідовність - ***\d*** (від слова **digit**). За змістом вона повністю ідентична ***[0-9]***, але коротше.

***PS C:\> Get-Process | Where-Object {$\_.Name -match "\d"}***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------- ----- ----- ----- ----- ------ -- -----------***

***93 10 1788 2336 70 1,25 548 FlashUtil10c***

***158 12 6500 1024 96 0,14 3336 smax4pnp***

***30 6 764 160 41 0,02 3920 TabTip32***

Так само послідовність була виділена для групи "будь-які літери будь-якого алфавіту, будь-які цифри, або символ підкреслення" ця група позначається як ***\w*** (від **word**) вона приблизно еквівалентна конструкції ***[a-zа-я\_0-9]*** (в ***\w*** ще входять символи інших алфавітів які використовуються для написання слів).

Інша популярна група: ***\s*** - "проміжок, або інший символ проміжку" (наприклад символ табуляції). Скорочення від слова **space**. У більшості випадків ви можете позначати проміжок просто як проміжок, але ця конструкція додає читабельності регулярному виразу.

Не менш популярною групою можна назвати символ ***.*** (**точка**). Точка в регулярних виразах є аналогічною за змістом знаку питання в підстановлювальних символах, тобто позначає один будь-який символ.

Всі перераховані вище конструкції можна використовувати як окремо, так і в складі груп, наприклад ***[\s \d***] буде відповідати будь цифрі або пробілу. Якщо ви хочете вказати всередині групи символ ***-*** (тире/мінус) то треба або екранувати його символом ***\*** (зворотний слеш), або поставити його на початку групи, щоб він не був випадково витлумачений як діапазон:

***PS C:\> "?????", "Word", "123", "-" -match "[-\d]"***

***123***

***-***

***Негативні групи і якоря***

Розглянемо деякі більш "просунуті" конструкції регулярних виразів.

Передбачається, що ви вже знаєте, як вказати регулярному виразу, які символи і/або їх послідовності повинні бути в рядку для збігу. А що якщо потрібно вказати не ті символи, які повинні бути присутніми, а ті яких не повинно бути? Тобто якщо потрібно вивести лише приголосні букви, ви можете їх перерахувати, а можете використовувати і негативну групу з голосними, наприклад:

***PS C:\> "a","b","c","d","e","f","g","h" -match "[^aoueyi]"***

***b***

***c***

***d***

***f***

***g***

***h***

"Кришка" в якості першого символу групи символів означає саме **заперечення**. Тобто на місці групи може бути присутнім будь-який символ окрім перерахованих в ній. Для того щоб включити заперечення в символьних групах (***\d***, ***\w***, ***\s***), не обов'язково укладати їх у квадратні дужки, досить перевести їх у верхній регістр. Наприклад ***\D*** буде означати "що завгодно, крім цифр", а ***\S*** "все крім проміжків"

***PS C:\> "a","b","1","c","45" -match "\D"***

***a***

***b***

***c***

***PS C:\> "a","-","\*","c","&" -match "\W"***

***-***

***\****

***&***

Символьні групи дозволяють вказати лише вміст однієї позиції, один символ, що знаходиться в невизначеному місці рядка. А що, якщо треба наприклад вибрати всі слова, які починаються з літери ***w***? Якщо просто помістити цю букву в регулярний вираз, то воно співпаде для всіх рядків, де ***w*** взагалі зустрічається, і не важливо - на початку, в середині або в кінці рядка. У таких випадках на допомогу приходять "якоря". Вони дозволяють проводити порівняння, починаючи з певної позиції в рядку.

***^*** (Кришка) є якорем початку рядка, а ***$*** (знак долара) - позначає кінець рядка.

Не заплутайтесь - ***^*** як символ заперечення використовується лише на початку групи символів, а поза групою - цей символ є вже якорем. Авторам регулярних виразів явно не вистачало спеціальних символів, і вони по можливості використовували їх більш ніж в одному місці.

*Приклад.* Виведення списку процесів, імена яких починаються з літери ***w***:

***PS C:\> Get-Process | Where-Object {$\_.Name -match "^w"}***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------- ------ ----- ----- ----- ------ -- -----------***

***80 10 1460 156 47 0,11 452 wininit***

***114 9 2732 1428 55 0,56 3508 winlogon***

***162 11 3660 1652 44 0,14 3620 wisptis***

***225 20 5076 4308 95 31,33 3800 wisptis***

Ця команда вивела процеси, у яких відразу після початку імені (***^***) слідує символ ***w***. Інакше кажучи, ім'я починається на ***w***. Для ускладнення прикладу, і для спрощення розуміння, додамо сюди "кришку" в значенні негативної групи:

***PS C:\> Get-Process | Where-Object {$\_.Name -match "^w[^l-z]"}***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------- ------ ----- ----- ----- ------ -- -----------***

***80 10 1460 156 47 0,11 452 wininit***

***114 9 2732 1428 55 0,56 3508 winlogon***

***162 11 3660 1652 44 0,14 3620 wisptis***

***225 20 5076 4308 95 31,33 3800 wisptis***

Тепер команда вивела процеси, у яких ім'я починається з символу ***w***, а наступний символ є чим завгодно, тільки не символом з діапазону ***l-z***.

Для закріплення випробуємо другий якір - кінець рядка:

***PS C:\> "Яблука", "Груші", "Диня", "Єнот", "Апельсини", "Персик" -match "[аиі]*** ***$"***

***Яблука***

***Груші***

***Апельсини***

Цей вислів вивів нам все слова в яких остання буква ***а***, ***и*** або ***і***.

Якщо ви можете точно описати вміст всього рядка, то ви можете використовувати і обидва якоря водночас:

***PS C: \> "abc", "adc", "aef", "bca", "aeb", "abec", "abce" -match "^a.[сb]$"***

***abc***

***adc***

***aeb***

Це регулярний вираз виводить всі рядки, які починаються з літери ***а***, за якою слідує один будь-який символ (точка), потім символ ***c*** або ***b*** і потім кінець рядка.

Позначення деяких класів символів (метасимволів) наведені в табл. 3

Табл. 3. Метасимволи, використовувані в регулярних виразах

|  |  |
| --- | --- |
| **Метасимвол** | **Опис метасимвола** |
| ***.(точка)*** | Припускає, що в кінцевому виразі на її місці буде стояти будь-який символ. Продемонструємо це на прикладі набору англійських слів:  *Вихідний набір рядків:*  ***wake***  ***make***  ***machine***  ***cake***  ***maze***  *Регулярний вираз:*  ***ma.e***  *Результат:*  ***make***  ***maze*** |
| ***\w*** | Заміщує будь-які символи, які відносяться до букв, цифр і знаку підкреслення. Приклад:  *Вихідний набір рядків:*  ***abc***  ***а$c***  ***a1c***  ***аc***  *Регулярний вираз:*  ***a\wc***  *Результат:*  ***abc***  ***a1c*** |
| ***\W*** | Заміщує всі символи, крім букв, цифр і знака підкреслення (тобто є зворотним метасимвол ***\w***). Приклад:  *Вихідний набір рядків:*  ***abc***  ***а$с***  ***a1c***  ***а с***  *Регулярний вираз:*  ***a\Wc***  *Результат:*  ***а$с а с*** |
| ***\d*** | Заміщує всі цифри. Продемонструємо його дію на тому ж прикладі:  *Вихідний набір рядків:*  ***abc***  ***а$с***  ***a1c***  ***а с***  *Регулярний вираз:*  ***a\dc***  *Результат:*  ***a1c*** |
| ***\D*** | Заміщує всі символи, крім цифр, наприклад:  *Вихідний набір рядків:*  ***abc***  ***а$с***  ***a1c***  ***а с***  *Регулярний вираз:*  ***a\Dc***  *Результат:*  ***abc***  ***а$с***  ***а с*** |

***Завдання***

1. Вивести список запущених служб, назва яких починається на слова “***MS***” (формат виведення обрати на власний розсуд).
2. Знайти у каталозі ***С:\Windows\Microsoft.NET*** та всіх його підкаталогах файли динамічних бібліотек (***.dll***), назви яких починаються на “***aspnet***”.
3. Знайти у кореневому каталозі Вашого мережевого диску та його підкаталогах файли зображень, що мають розширення ***.jpg*** та містять у своїй назві тільки літери українського алфавіту.
4. Знайти у кореневому каталозі Вашого мережевого диску та його підкаталогах файли, у імені яких у якості другого символу використовується літера “***a***”
5. Знайти у каталозі ***С:\Windows*** конфігураційні файли, в іменах яких присутні числа.
6. Вивести список команд, які тим чи іншим чином стосуються роботи з процесами (їх імена закінчуються на слово “***Process***”).

***Контрольні питання***

1. Для чого призначені регулярні вирази?
2. Які дії виконує оператор ***–match*** у **PowerShell**? У чому різниця операторів ***–match*** та ***–cmatch***?
3. Яким чином задаються групи символів у регулярних виразах?
4. Які сиволи у регулярних виразах позначають відповідно початок та кінець рядка?
5. Які метасимволи використовуються у регулярних виразах PowerShell? Наведіть приклади.